

# **MODIS**

# **SCIENCE DATA SUPPORT TEAM**

# **PRESENTATION**

**January 3, 1992**

## **AGENDA**

1. Action Items
2. TEG Status
3. Strawman SDST Work Plan Reports

ACTION ITEMS:

08/30/91 [Lloyd Carpenter and Team]: Draft a schedule of work for the next 12 months. Include primary events and milestones, documents to be produced, software development, MAS support, etc. (The work plan in Microsoft Project has been updated. Sample reports are included in the handout.) STATUS: Open. Due date 09/27/91.

12/06/91 [Liam Gumley]: Investigate a cataloguing scheme for the MAS data. Consider the Master Catalogue, PLDS and PCDS. STATUS: Open. Due date 02/14/92.

12/06/91 [Liam Gumley, Tom Goff, Ed Masuoka]: Develop a plan for storing and distributing MAS data. STATUS: Open. Due date 02/14/92.

**Tom Goff's Status  
for  
2 January, 1992**

TGoff on GSFC mail,  
[teg@LTPIRIS2.GSFC.NASA.GOV](mailto:teg@LTPIRIS2.GSFC.NASA.GOV),  
or (301) 982-3704

- \* **Code Portability** - The file dumping utility that was mentioned last time has been ported to several machines and the source listing included with this handout. I have included this handout to be used as a illustration of the automatic documentation facilities that I am in the process of developing. Any and all comments are appreciated.
- \* **Anonymous FTP facilities** - Liam and I have been and will continue to post files and data sets to the anonymous ftp file site on the ltp iris computer. This file site is currently owned by "Gumley" thereby preventing myself or others from posting or updating files. This will be fixed in the future but brings up the problem of allowing free access to computer files versus a reasonable amount of computer protection.

**DISCUSSION ITEMS**

- \* The desired capability for a proposed MODIS SDST bulletin board.

```

/*h           FDUMP.C  a Generic File Dumping Utility          */
/*d
This program will dump the contents of a file in any user specified
format. The beginning Byte location within the file and the length
of the dump can be specified. The Bytes within the file can be
displayed in ASCII, EBCDIC, hexadecimal, octal, integer (16 and
32 bit), floating point (32 or 64 bit), or a special mnemonic
format. This is handy for finding characters that are not normally
displayable, such as <CR> for carriage returns or <XOFF> for the
xoff (control-s) character. See also: the companion program: replace.c
to fix up these undisplayable characters.                                */

/*a
Written by:
Thomas E. Goff, Research and Data Systems Corporation
(301) 982-3704
teg@ltpiris2.gsfc.nasa.gov                                         */

/*r Version: revision 1.0, 2 January, 1992                         */

/*u USAGE:
fdump -<options> +<start_Byt> #<how_many_Bytes> <file_name>
where: <options> can optionally be
      h for a hex dump (default)
      d for a decimal dump of each Byte
      o for an octal dump of each Byte
      a for an ASCII dump
      e for an EBCDIC dump
      m for the special mnemonic dump
      i16 for a 16 bit integer dump (2 Bytes)
      i32 for a 32 bit integer dump (4 Bytes)
      f32 for a 32 bit floating point dump (4 Bytes)
      f64 for a 64 bit floating point dump (8 Bytes)
      <start_byte> is the optional offset from the beginning of
          the file to begin dumping (o is the first Byte)
      <how_many_Bytes> is the optional number of Bytes to dump
and <file_name> is the file to be dumped

/*t  This program has been tested on the following computers:
SGA personal IRIS
DEC VAX VMS
PC clones with PowerC and Borland C++
Sun Sparc2                                         */

/*k  < key words > file; dump; display functions; file dump;          */
/*r  < revision history >
    TEG - initial release                                         */

/*b  < bugs / future enhancements >                                     */

#include <stdio.h>
#include <ctype.h>

#define HUGE 32767          /* the largest int number on this machine */
/* (use 32767 for PC's , 2147483647 otherwise) */
#define BufferSize 8640      /* This is a 'magic' number (n*2160) */
/* hint: lcd of {60,16,10,9,8,6,5,4} */

#define HEX 0x01
#define DEC 0x02
#define OCT 0x03
#define ASC 0x10
#define EBC 0x04
#define MNU 0x20
#define I16 0x30
#define I32 0x40
#define F32 0x50
#define F64 0x60

unsigned char e2a[256] = {      /* ASCII to EBCDIC conversion chart */
    0,   1,   2,   3, 156,   9, 134, 127, 151, 141, 142, 11,
    12,  13,  14,  15, 16, 17, 18, 19, 157, 133,   8, 135,
    24,  25, 146, 143, 28, 29, 30, 31, 128, 129, 130, 131,
   132, 10, 23, 27, 136, 137, 138, 139, 140,   5,   6,   7,
   144, 145, 22, 147, 148, 149, 150,   4, 152, 153, 154, 155,
}

```

```

20, 21, 158, 26, 32, 160, 161, 162, 163, 164, 165, 166,
167, 168, 91, 46, 60, 40, 43, 33, 38, 169, 170, 171,
172, 173, 174, 175, 176, 177, 93, 36, 42, 41, 59, 94,
45, 47, 178, 179, 180, 181, 182, 183, 184, 185, 124, 44,
37, 95, 62, 63, 186, 187, 188, 189, 190, 191, 192, 193,
194, 96, 58, 35, 64, 39, 61, 34, 195, 97, 98, 99,
100, 101, 102, 103, 104, 105, 196, 197, 198, 199, 200, 201,
202, 106, 107, 108, 109, 110, 111, 112, 113, 114, 203, 204,
205, 206, 207, 208, 209, 126, 115, 116, 117, 118, 119, 120,
121, 122, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219,
220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231,
123, 65, 66, 67, 68, 69, 70, 71, 72, 73, 232, 233,
234, 235, 236, 237, 125, 74, 75, 76, 77, 78, 79, 80,
81, 82, 238, 239, 240, 241, 242, 243, 92, 159, 83, 84,
85, 86, 87, 88, 89, 90, 244, 245, 246, 247, 248, 249,
48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 250, 251,
252, 253, 254, 255};

char *mneu[] = {
    /* ASCII to Mnemonic character array */
    "<NUL>", "<SOH>", "<STX>", "<ETX>", "<EOT>",
    "<ENQ>", "<ACK>", "<BEL>", "<BS>", "<HT>",
    "<LF>", "<VT>", "<FF>", "<CR>", "<SO>",
    "<SI>", "<DLE>", "<XON>", "<DC2>", "<XOFF>",
    "<DC4>", "<NAK>", "<SYN>", "<ETB>", "<CAN>",
    "<EM>", "<SUB>", "<ESC>", "<FS>", "<GS>",
    "<RS>", "<US>", "!", "\\", "#", "$", "%", "&", "\",
    "\\", "\\", "\\", "+", "\\", "\\", "\\", "\\", "\\", "\",
    "2", "3", "4", "5", "6", "7", "8", "9", ":", ",",
    "<", "=", ">", "?", "Q", "A", "B", "C", "D", "E",
    "F", "G", "H", "I", "J", "K", "L", "M", "N", "O",
    "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y",
    "Z", "[", "\\", "]", "\^", "\\", "\\", "a", "b", "c",
    "d", "e", "f", "g", "h", "j", "k", "l", "m",
    "n", "o", "p", "q", "r", "s", "t", "u", "v", "w",
    "x", "y", "z", "{", "|", "}", "~", "<DEL>"};

union {
    /* file contents data structure */
    unsigned char buff[BufferSize];
    short sbuff[BufferSize/2];
    long lbuff[BufferSize/4];
    float fbuff[BufferSize/4];
    double dbuff[BufferSize/8];
} *p;

main(ac,av)
int ac;                      /* the number of command arguments */
char **av;                     /* array of command arguments */
{
    char filename[256];        /* the file to be dumped */
    FILE *in;                  /* system file pointer */
    char flag;                 /* user requested dump format */
    int argf;                  /* number of arguments sucessfully parsed */
    int startB = 0;             /* user requested starting Bytes offset */
    int numB = 0;                /* user requested Bytes to dump */
    register int i;              /* local generic index */

    flag = 0x00;

    if(ac == 1) {               /* check for only one argument */
        usage_and_die();
        exit(-1);
    }

    flag |= HEX;

    argf=1;
    for( i=1; i<ac; i++) {
        if(av[i][0] == '-') {      /* look for options */
            ++argf;
            switch (av[i][1]) {
                case 'h': flag = HEX;
                    break;
                case 'd': flag = DEC;
                    break;
                case 'o': flag = OCT;
                    break;
            }
        }
    }
}

```

```
        break;
    case 'a': flag = ASC;
        break;
    case 'e': flag = EBC;
        break;
    case 'm': flag = MNU;
        break;
    case 'i': if((av[i][2] == '1') && (av[i][3] == '6')) {
        flag = I16 ;
        break;
    }
        if((av[i][2] == '3') && (av[i][3] == '2')) {
        flag = I32 ;
        break;
    }
    case 'f': if((av[i][2] == '3') && (av[i][3] == '2')) {
        flag = F32 ;
        break;
    }
        if((av[i][1] == 'f') && (av[i][2] == '6') && (av[i][3] == '4')) {
        flag = F64 ;
        break;
    }

default: printf("\n*** Unknown option: %s ??\n",av[1]);
    usage_and_die();
    break;
}
/* check for illegal option combinations */
if((av[i][2] == 'e') && (av[i][3] == 0) && (flag & 0x03)) flag |= EBC;
else if (av[i][2] != 0 && av[i][2] != 'a' && flag != I16 &&
    flag != I32 && flag != F32 && flag != F64) {
    printf("\n*** Illegal combination: %s ??\n",av[1]);
    usage_and_die();
}
}
if(av[i][0] == '+') /* check for starting Byte request */
++argf;
if(sscanf(av[i]+1,"%d",&startB) == 0) {
    printf("\n*** Invalid starting Byte specified\n");
    usage_and_die();
}
if(startB < 0) {
    printf("\n*** +%d must be a positive number\n",startB);
    usage_and_die();
}
}
if(av[i][0] == '#') /* check for Byte dump limit */
++argf;
if(sscanf(av[i]+1,"%d",&numB) == 0) {
    printf("\n*** Invalid number of Bytes specified\n");
    usage_and_die();
}
if(numB < 0) {
    printf("\n*** #%d must be a positive number\n",numB) ;
    usage_and_die();
}
}
}
if((ac-argf) > 1) /* check for valid file specification */
printf("\n*** Only one file at a time - please\n");
usage_and_die();
strcpy(filename,av[ac-1]);

if((flag & ASC) && (flag & EBC)) {
    printf("\n*** Cannot do ASCII & EBCDIC together\n");
    usage_and_die();
}

in = fopen(filename,"rb"); /* (file must be opened in binary mode) */
if(in == NULL) {
    printf("\nCannot open: '%s' for read. Exiting...\n",filename);
    exit(-1);
}
```

```
xdump(in,flag,startB,numB);
fclose(in);
}

/*
usage_and_die()
{
printf("\nUsage: fdump [-h|d|o|a|e|m|i16|i32|f32|f64, [+nnn [#nnn]]] <file>\n");
  printf("      -h = hex dump (default)\n");
  printf("      -d = decimal dump\n");
  printf("      -o = octal dump\n");
  printf("      -a = ASCII dump\n");
  printf("      -e = EBCDIC dump\n");
  printf("      -m = mnemonic ASCII dump\n");
  printf("      -i16 = 'short' conversion\n");
  printf("      -i32 = 'long' conversion\n");
  printf("      -f32 = 'float' conversion\n");
  printf("      -f64 = 'double' conversion\n");
  printf("      +nnn = starting offset in Bytes\n");
  printf("      #nnn = length of dump in Bytes\n\n");
  printf(" Note: d/h/o include an ASCII dump,\n");
  printf(" de/he/oe include an EBCDIC dump,\n");
  printf(" a and e CANNOT be used together,\n");
  printf(" and -option precedence applies\n");
  exit(-1);
}

/*
xdump(fp, flag, startB, numB)
FILE *fp;           /* file pointer */
char flag;          /* dump type */
int startB;         /* starting Byte offset */
int numB;           /* number of Bytes to dump */
{
char str[100];      /* first output string */
char chr[100];      /* second output string */
char tmp[50];        /* temporary string */
register int i;      /* indices */
register int j;      /* indices */
register int offset; /* corrected starting Byte offset */
int step;            /* number of items per CRT line */
int numread;         /* number of Bytes from file read */
int len;              /* length in Bytes of input buffer */
int Bpw = 1;          /* number of Bytes per word */

/* Let's do the listing headers first */
if((flag & 0x03) == HEX) {
  step = 16;
  sprintf(str," Offset             Hex Value           ");
  if(flag & EBC) strcat(str,"EBCDIC\n");
  else strcat(str,"ASCII\n");
  sprintf(chr," ----- ----- ----- ----- \n");
}
if((flag & 0x03) == DEC) {
  step = 10;
  sprintf(str," Offset             Decimal Value       ");
  if(flag & EBC) strcat(str,"EBCDIC\n");
  else strcat(str,"ASCII\n");
  sprintf(chr," ----- ----- ----- ----- \n");
}
if((flag & 0x03) == OCT) {
  step = 8;
  sprintf(str," Offset             Octal Value        ");
  if(flag & EBC) strcat(str,"EBCDIC\n");
  else strcat(str,"ASCII\n");
  sprintf(chr," ----- ----- ----- ----- \n");
}
if(flag == ASC) {
  step = 60;
  sprintf(str," Offset             ASCII Value\n");
  sprintf(chr," ----- ----- \n");
}
```

```
if(flag == EBC) {
    step = 60;
    sprintf(str,"  Offset           EBCDIC Value\n");
    sprintf(chr," -----");
}
if(flag == MNU) {
    step = 1;
    sprintf(str,"  Offset           Mnemonic ASCII equivalents\n");
    sprintf(chr," -----");
}
if(flag == I16) {
    step = 9; Bpw = 2;
    sprintf(str,"  Offset           16 bit integers\n");
    sprintf(chr," -----");
}
if(flag == I32) {
    step = 5; Bpw = 4;
    sprintf(str,"  Offset           32 bit integers\n");
    sprintf(chr," -----");
}
if(flag == F32) {
    step = 6; Bpw = 4;
    sprintf(str,"  Offset           32 bit floats\n");
    sprintf(chr," -----");
}
if(flag == F64) {
    step = 4; Bpw = 8;
    sprintf(str,"  Offset           64 bit floats\n");
    sprintf(chr," -----");
}
printf("%s",str);
printf("%s",chr);
/* pull file reads until the starting Byte is found */
len = sizeof(p->buff);
p = (void *)malloc(BufferSize);
for(j=0; j<startB/len; j++) numread = fread(p->buff, sizeof(char), len, fp);
offset = startB-startB/len*len;
if(offset > 0)numread = fread(p->buff, sizeof(char), offset, fp);
if(startB > j*len+numread) {
    printf("\n*** The file is smaller than %d Bytes\n",startB);
    exit(2);
}
offset = startB;
for(j = 0; j < 100; j++) str[j] = 0;
/* now do the actual file dumping */
i=0 ; j=0 ; if(numB == 0)numB = HUGE;
if(flag == MNU) { /* the special Mnemonic dump */
    while(numB) {
        numB--;
        if(i == 0) {
            numread = fread(p->buff, sizeof(char), len, fp);
            if((numread < len) && (numread < numB))numB=numread-1;
        }
        if(j == 0) {
            sprintf(str,"%10d    ",offset);
            j += 15;
        }
        if(p->buff[i] & 0x80) {
            strcat(str,"^");
            j++;
        }
        strcat(str,mneu[p->buff[i]]);
        j += strlen(mneu[p->buff[i]]);
        if(j > 75) {
            printf("%s\n",str);
            for(j = 0; j < 100; j++) str[j]=0;
            j=0;
        }
        i++; if(i == len)i=0;
        offset++;
    }
    printf("%s\n",str);
}
```

```
>
else {                                /* all other file dumping types */
    /* (the type is determined by the step parameter) */
do {
    numread = fread(p->buff, sizeof(char), len, fp);
    if((numB != 0) && (numB < len)) numread = numB ;
    numB -= len;

    for(i = 0; i < numread/Bpw; i+=step) {

        switch (step) {
            case 16: sprintf(str,"%7x    ",offset);
                        break;
            case 4:
            case 5:
            case 6:
            case 9:
            case 10: sprintf(str,"%10d    ",offset);
                        break;
            case 8: sprintf(str,"%12o    ",offset);
                        break;
            case 60: sprintf(str,"%10d    ",offset);
                        break;
        }
        for(j = 0; (j < step) && (j+i < numread/Bpw); j++) { /* do values.. */
            switch (step) {
                case 16: sprintf(tmp,"%2x ",p->buff[i+j]);
                            strcat(str,tmp);
                            break;
                case 4: sprintf(tmp,"%14.6E ",p->dbuf[i+j]);
                            strcat(str,tmp);
                            break;
                case 5: sprintf(tmp,"%11d ",(long)p->lbuf[i+j]);
                            strcat(str,tmp);
                            break;
                case 6: sprintf(tmp,"%10.3e ",p->fbuf[i+j]);
                            strcat(str,tmp);
                            break;
                case 9: sprintf(tmp,"%6hd ",p->sbuf[i+j]);
                            strcat(str,tmp);
                            break;
                case 10: sprintf(tmp,"%3d ",p->buff[i+j]);
                            strcat(str,tmp);
                            break;
                case 8: sprintf(tmp,"%3o ",p->buff[i+j]);
                            strcat(str,tmp);
                            break;
                case 60: if(flag & EBC)
                            p->buff[i+j] = e2a[p->buff[i+j]];
                            if(isprint(p->buff[i+j])) {
                                sprintf(tmp,"%c",p->buff[i+j]);
                                strcat(str,tmp);
                            }
                            else
                                strcat(str,".");
                            break;
            }
        }
        for(; j < step; j++)
            switch (step) {
                case 16: strcat(str,"    ");
                            break;
                case 10: strcat(str,"    ");
                            break;
                case 8: strcat(str,"    ");
                            break;
            }
        switch (step) {
            case 16: strcat(str,"  ");
                        break;
            case 10: strcat(str,"  ");
                        break;
            case 8: strcat(str,"  ");
                        break;
        }
    }
}
```

```
        break;
    }

    for(j = 0; (j < step) && (j+i < numread); j++) { /* do ascii */
        if(step == 8 || step == 10 || step == 16) {
            if(flag & EBC)
                p->buff[i+j] = e2a[p->buff[i+j]];
            if(isprint(p->buff[i+j])) {
                sprintf(tmp,"%c",p->buff[i+j]);
                strcat(str,tmp);
            }
            else
                strcat(str,".");
        }
        printf("%s\n",str);
        offset += step;
    }
}
while(numread/Bpw == sizeof(p->buff));
}
```

### Activity Detail Report

Project: FY92001

Date: Jan 2, 1992 3:12 PM

2 Draft List of Algorithms

Early Start: Dec 2, 1991 8:30 AM  
Early Finish: Jan 29, 1992 5:30 PM

Duration: 8.00 Weeks

Late Start: Dec 26, 1991 1:30 PM  
Late Finish: Feb 25, 1992 1:30 PM

Slack: 17.50 Days

Sched Start: Dec 26, 1991 1:00 PM  
Sched Finish: Feb 25, 1992 1:00 PM

#### Notes:

NONE

#### Resources Allocated:

Name	Duration (days)	Amount used	Cost	Cost Basis	Cost to Complete
Al McKay	40.00	0.50	\$0.00	Hour	\$0.00
Tom	40.00	0.50	\$0.00	Hour	\$0.00
Total:					\$0.00

#### Predecessors:

NONE

#### Successors:

4 Estimate time for each step  
Slack: 2.44 Days

Early Start: Feb 21, 1992 9:00 AM  
Late Start: Feb 25, 1992 1:30 PM

Table

Project: FY92001

Date: Jan 2, 1992 3:12 PM

1 TM S/W DEVELOPMENT

Early Start: Dec 2, 1991 8:30 AM	Early Finish: Dec 2, 1991 8:30 AM
Late Start: Oct 6, 1992 5:00 PM	Late Finish: Oct 6, 1992 5:00 PM
Sched Start: Oct 6, 1992 5:00 PM	Sched Finish: Oct 6, 1992 5:00 PM

Duration: 0.00 Days

Slack: 211.00 Days

Predecessors: None

Successors: None

Resource: None

2 Draft List of Algorithms

Early Start: Dec 2, 1991 8:30 AM	Early Finish: Jan 29, 1992 5:30 PM
Late Start: Dec 26, 1991 1:30 PM	Late Finish: Feb 25, 1992 1:30 PM
Sched Start: Dec 26, 1991 1:00 PM	Sched Finish: Feb 25, 1992 1:00 PM

Duration: 8.00 Weeks

Slack: 17.50 Days

Predecessors: None

Successors: 4

Resource: Al McKay, Tom

3 Identify Steps

Early Start: Dec 2, 1991 8:30 AM	Early Finish: Dec 20, 1991 5:30 PM
Late Start: Sep 10, 1992 8:30 AM	Late Finish: Sep 30, 1992 5:30 PM
Sched Start: Sep 10, 1992 8:30 AM	Sched Finish: Sep 30, 1992 5:30 PM

Duration: 3.00 Weeks

Slack: 196.00 Days

Predecessors: None

Successors: None

Resource: Lloyd

	Resources	Period	Demand
Project: FY92001			Date: Jan 2, 1992 3:12 PM
Time scale: Week			Demand scale: Hour
Period ending	Mar 23, 1992	Mar 30, 1992	Apr 6, 1992
1 Al McKay	60.00	60.00	60.00
2 Tom	64.00	64.00	64.00
3 Lloyd	48.00	48.00	48.00
4 Liam	40.00	40.00	42.40
Total:	----- 212.00	----- 212.00	----- 214.40

### Resources Detail

Project: FY92001  
Resource: Al McKay  
Amount used: 209.25 Days

Date: Jan 2, 1992 3:12 PM  
Unit Cost: \$0.00 / Hour  
Cost to complete: \$0.00

#### # Activity:

##### 2 Draft List of Algorithms

Duration: 8.00 Weeks  
Slack: 17.50 Days  
Amount used: 0.50  
Cost: \$0.00

Early Start: Dec 2, 1991 8:30 AM  
Early Finish: Jan 29, 1992 5:30 PM  
  
Late Start: Dec 26, 1991 1:30 PM  
Late Finish: Feb 25, 1992 1:30 PM  
  
Sched Start: Dec 26, 1991 1:00 PM  
Sched Finish: Feb 25, 1992 1:00 PM

##### 4 Estimate time for each step

Duration: 6.00 Weeks  
Slack: 2.44 Days  
Amount used: 0.50  
Cost: \$0.00

Early Start: Feb 21, 1992 9:00 AM  
Early Finish: Apr 3, 1992 9:00 AM  
  
Late Start: Feb 25, 1992 1:30 PM  
Late Finish: Apr 7, 1992 1:30 PM  
  
Sched Start: Feb 25, 1992 1:30 PM  
Sched Finish: Apr 7, 1992 1:30 PM

##### 7 Identify Prototype Algorithms

Duration: 4.00 Weeks  
Slack: 2.44 Days  
Amount used: 1.00  
Cost: \$0.00

Early Start: Jun 1, 1992 9:00 AM  
Early Finish: Jun 29, 1992 9:00 AM  
  
Late Start: Jun 3, 1992 1:30 PM  
Late Finish: Jul 1, 1992 1:30 PM  
  
Sched Start: Jun 3, 1992 1:30 PM  
Sched Finish: Jul 1, 1992 1:30 PM

##### 8 SDST Prototype Action

Duration: 12.70 Weeks  
Slack: 2.44 Days  
Amount used: 0.50  
Cost: \$0.00

Early Start: Jun 29, 1992 9:00 AM  
Early Finish: Sep 28, 1992 2:00 PM  
  
Late Start: Jul 1, 1992 1:30 PM  
Late Finish: Sep 30, 1992 5:30 PM  
  
Sched Start: Jul 1, 1992 1:30 PM  
Sched Finish: Sep 30, 1992 5:30 PM